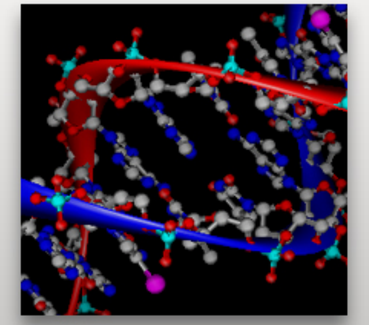
Core OS Enhancements for BSD Developers

Session 510

These are confidential sessions—please refrain from filming or taking pictures.



OS Foundations

Core OS Enhancements for BSD Developers

Marc Majka
Senior Software Engineer

http://developer.apple.com/wwdc2005/

Apple System Logger (ASL)

What Is ASL?

- ASL is a new approach to creating and managing system log messages in Mac OS X
- New and modified components
  - New asl(3) logging API
  - Changes in syslog(3) library code
  - Rewritten syslogd server
  - New syslog command-line utility

Goals of ASL

- Structured and flexible message format
- Reduce log file proliferation
- Facilitate viewing and searching

Messages With BSD syslog

- Priority levels 0 (Emergency)–7 (Debug)
- Fix set of Facilities
- Fixed format

```
Jun 24 12:03:17 zippy myapp[945]: Hello World
```

Messages With ASL

- Extensible key/value dictionary
- Keys and Values are null-terminated strings
- Standard keys
  - Level, Time, Sender, PID, Message…
- Additional keys appropriate to your application
  - e.g. Security, Language, Color…

```
Time    2005.06.07 12:03:17 UTC
Host    zippy
Sender  myapp
PID     945
UID     501
GID     80
Level   6
Message Processing request for new data
```

Message API

Simple Example

```
asl_log(NULL, NULL, ASL_LEVEL_DEBUG, "Hello World");
```

```
Time       2005.06.07 12:03:52 UTC
Host       zippy
Sender     myapp
PID        945
UID        501
GID        80
Level      7
Facility   user
Message    Hello World
```

Message API

Advanced example

```
aslclient a;
aslmsg m;

a = asl_open(NULL, "MyFacility", 0);
m = asl_new(ASL_TYPE_MSG);

asl_set(m, "Subsystem", "Greeting");
asl_set(m, "Language", "English");

asl_log(a, m, ASL_LEVEL_INFO, "Hello World");

 asl_free(m);
 asl_close(a);
```

Advanced Example

Output

```
Time            2005.06.07 12:04:29 UTC
Host            zippy
Sender          myapp
PID             945
UID             501
GID             80
Level           6
Facility        MyFacility
Subsystem       Greeting
Language        English
Message         Hello World
```

Finding and Reading Messages

What ASL does with received messages

- Unified log message data store
- Search API to retrieve from the data store
- New “syslog” command-line tool for searching and monitoring messages being written to the data store
- Legacy support for traditional configuration and output
  - /etc/syslog.conf
  - /var/log/system.log et al.

Search API Example

```
aslclient a;
aslmsg q, m;
aslresponse r;

a = asl_open(NULL, NULL, 0);
q = asl_new(ASL_TYPE_QUERY);

asl_set_query(q, "Flavor", "Vanilla", ASL_QUERY_OP_EQUAL);

r = asl_search(a, q);
while (NULL != (m = aslresponse_next(r))
    ...

aslresponse_free(r);
asl_free(q);
asl_close(a);
```

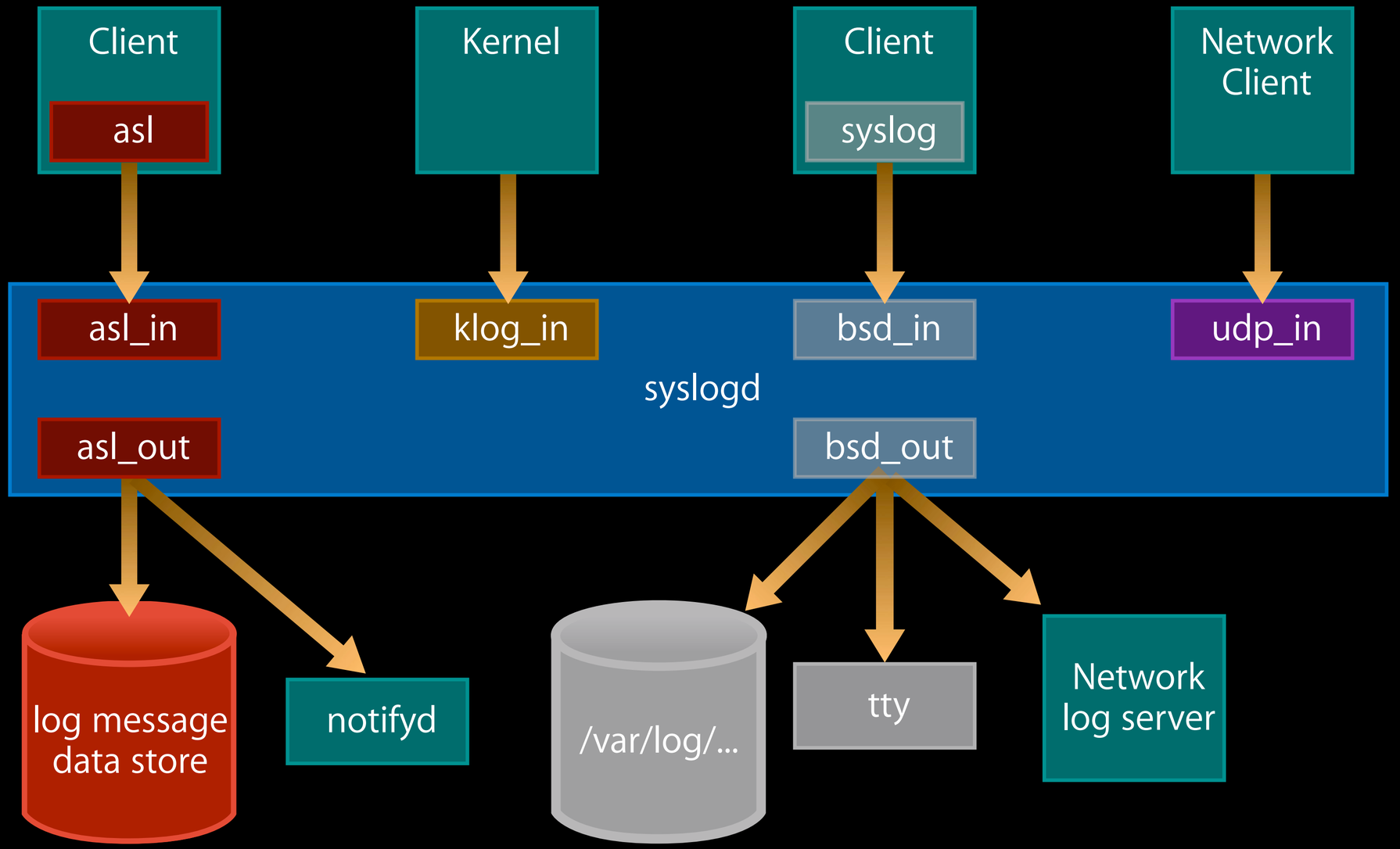
Searching From the Command Line

- syslog command line utility has many search options
- see the syslog(1) man page or “syslog -help”

```
syslog
syslog -k Flavor Vanilla
syslog -k Facility user -k Time ge -1d
syslog -w -F ‘$Time $(PID): $Message’ -k Level le 3
```

ASL Architecture

- syslogd is the ASL server
  - Input modules receive messages on various input channels
  - Output modules sort, file, and forward messages
- API for sending messages
  - <asl.h> (in libSystem)
  - <syslog.h>
  - kernel printf
- API for searching
  - <asl.h>
- syslog command-line tool for searching and monitoring


Client
asl
Kernel
Client
syslog
Network Client
asl_in
klog_in
bsd_in
udp_in
syslogd
asl_out
bsd_out
log message data store
notifyd
/var/log/...
tty
Network log server

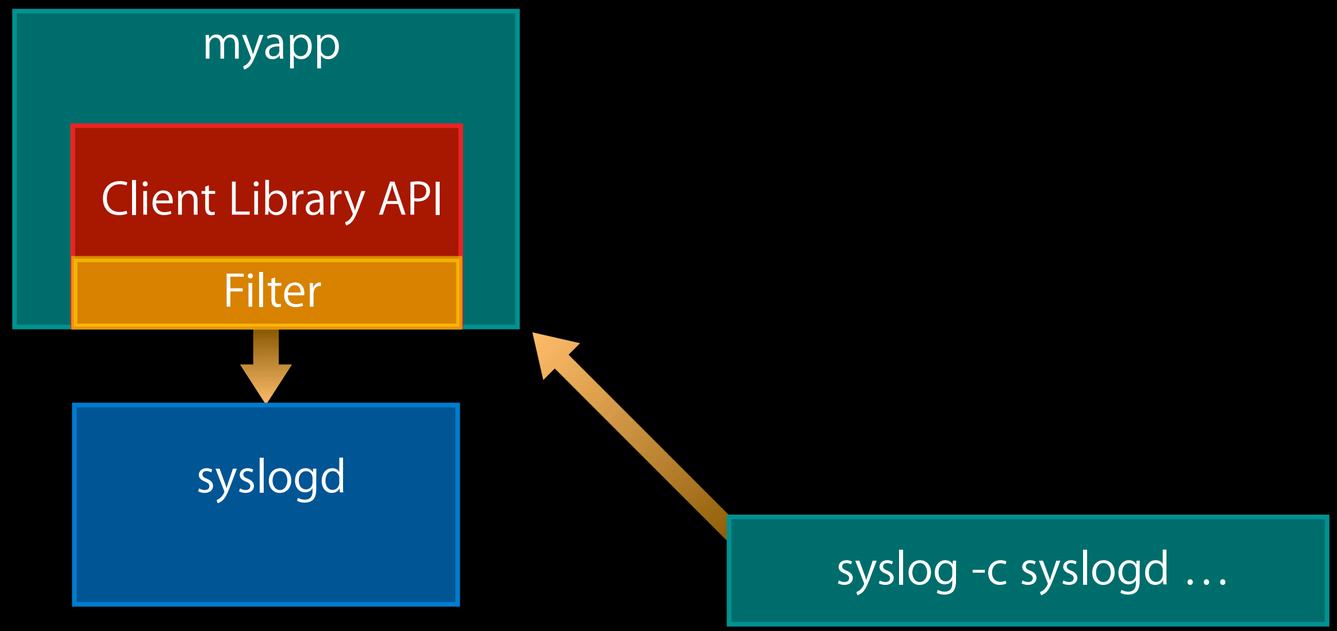
Filter Controls

Regulating the flow and storage of messages

- Client library filtering controls which priority levels are sent to syslogd
- Server filter controls which priority levels are saved in the data store
- Remote control mechanism for adjusting filters

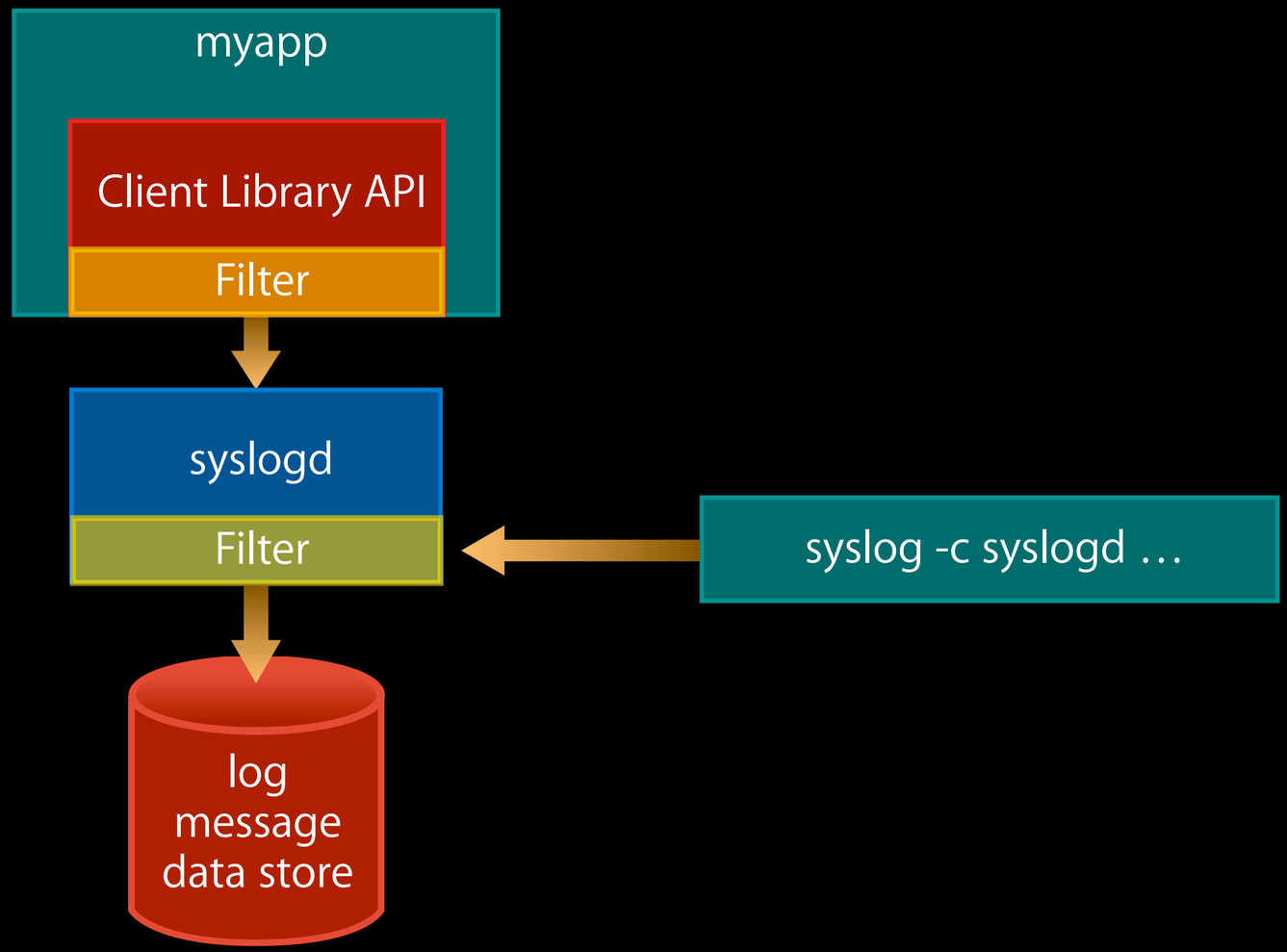
Filtering in the Client API

Controlling which messages are sent



Filtering in syslogd

Controlling what is saved in the data store



Local, Master, and Application Filters

- Local filter is set using asl_set_filter() or setlogmask()
- Master filter is normally off and has no effect
  - If master filter is set, it overrides local filter
  - Set using “syslog -c 0…”
- Application-specific filter overrides master and local filters
  - Set using “syslog -c myapp…”

Pruning the Data Store

- Default Data Store filter saves Emergency through Notice
- Daily cron script prunes data store
  - See /etc/periodic/daily/500.daily
- Use syslog -p to prune manually
- Data store improvements in the future

Service Management in Mac OS X

Dave Zarzycki
Senior Software Engineer

http://developer.apple.com/wwdc2005/

Introducing: launchd

The unification of background process management

- XML instead of shell scripts
- Less work for software developers
- Easy to use for simple scenarios
- Increased flexibility
- New kinds of launch-on-demand criteria
- Simplified notion of dependancies
- Support for user supplied jobs

XML

Structured data is a good thing

- Easy to introspect attributes of all background jobs
- Easy to change attributes of any background job
- Consistency

Less Work

No need to daemonize

- fork() and have the parent exit()
- setsid()
- Closing stray file descriptors
- Reopening stdio as /dev/null
- Changing the working directory to /
- etc.

Easy to Use

```
<key>Label</key>
<string>com.example.somethingd</string>
<key>OnDemand</key>
<false/>
<key>ProgramArguments</key>
<array>
                <string>/usr/sbin/somethingd</string>
</array>
```

Increased Flexibility

Standardized schema for control over how a daemon is started

- Standard out and error
- Nice level
- Working directory
- Root directory
- Umask
- Environment variables

Launch on Demand

Helping you help us save system resources

- Sockets (IPv4, IPv6 and Local/Unix domain)
- When file system objects change
- While queue directories contain files
- Interval timers
- Calendar-based interval timers

Dependancies

The old way

- SystemStarter used explicit out-of-band dependancies
- In reality, this didn't work. Often, one of the following happened:
  1. Dependancies were overstated
  2. Dependancies were understated
  3. Dependancy meaning was vague
  4. Dependancies were assumed by virtue of other dependancies (thus making re-factoring code hard)

Dependancies

The new way

- launchd uses implicit inline dependancies
- We register all sockets of all daemons first
- Then we start all non-demand-based daemons
- As daemons startup, they may talk to other daemons freely
- As daemons startup, they dequeue messages when they become ready

Dependancies Summary

Declare your sockets!

- Daemon writers must define their sockets in their XML plist to allow others to depend on their presence

Case Study

ssh-agent

- Complicated to automate the management of
- Most users only want one per session
- launchd makes this trivial with small modifications to ssh-agent
- http://www.opendarwin.org/~landonf/misc/launchd-sshagent/

Going Native With launchd

The high level overview

- A simple IPC API
- A simple RTTI-based object system to support message passing

C APIs

```
#include <launch.h>

launch_data_t launch_msg(launch_data_t);

launch_data_t launch_data_new_integer(long long);

long long launch_data_get_integer(launch_data_t);
```

C API Semantics

- launch_data_t represents an object graph
- launch_msg() is a synchronous API for the common case
  - Returns NULL and sets errno on failure
- If you request asynchronous messages be sent back
  - Call launch_msg(NULL) to get an asynchronous message
  - Keep calling until you get NULL back
  - If errno == 0, then no more asynchronous messages are available for reading

launch_data_t

RTTI and container classes are fun!

- Dictionaries
- Arrays
- File Descriptors
- Integers
- Real numbers
- Booleans
- Strings
- Opaque Data

launch_data_t APIs

"Just enough for IPC, and no more"

- Get/set operations for basic types
- Dictionaries:
  - insert
  - lookup
  - remove
  - iterate
- Arrays:
  - set index
  - get index
  - get count

XML plist Keys

Required

- Label
- ProgramArguments

Should have

- OnDemand

Optional

- UserName
- GroupName
- ProgramName
- Root
- Umask
- WorkingDirectory
- ProgramArguments
- EnvironmentVariables
- Sockets
- etc.

Sockets

Details, details, details…

- How to setup a given socket
- Who to connect to…
- Where to listen…
- What options to use...
- etc.

XML plist → launch_data_t

Data distillation

- Sockets → file descriptors
- Makes launchd unaware of socket types
- Future proofs launchd

Example Messages

Dictionaries where the key is the command

- SubmitJob
- RemoveJob
- GetJobs
- GetJob
- CheckIn
- Start
- Stop

Example

```
launch_data_t msg = launch_data_alloc(LAUNCH_DATA_DICTIONARY);
launch_data_t resp;

launch_data_dict_insert(msg,
        launch_data_new_string("com.example.hellod"),
        LAUNCH_KEY_STARTJOB);

resp = launch_msg(msg);

launch_data_free(msg);
launch_data_free(resp);
```

Rehash

launchd is the future

- Less work for you
  - "pre-daemonized" when your main() is called
  - Just check-in and go
  - Automatic restarting
- More flexibility in what event sources you react to
  - Multiple Unix file descriptors
- User agents
  - A powerful concept for per session background processes

Demo

Dave Zarzycki
Senior Software Engineer

http://developer.apple.com/wwdc2005/

More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

Who to Contact

Craig Keithley
I/O Technology Evangelist
keithley@apple.com.com

Q&A

Jordan Hubbard, Engineering Manager, BSD Technology Group
Dave Zarzycki, Senior Software Engineer
Marc Majka, Senior Software Engineer

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference